Intel® Itanium® Processor Abstraction Layer

Tony Luck
Principal Engineer
April 18th, 2007

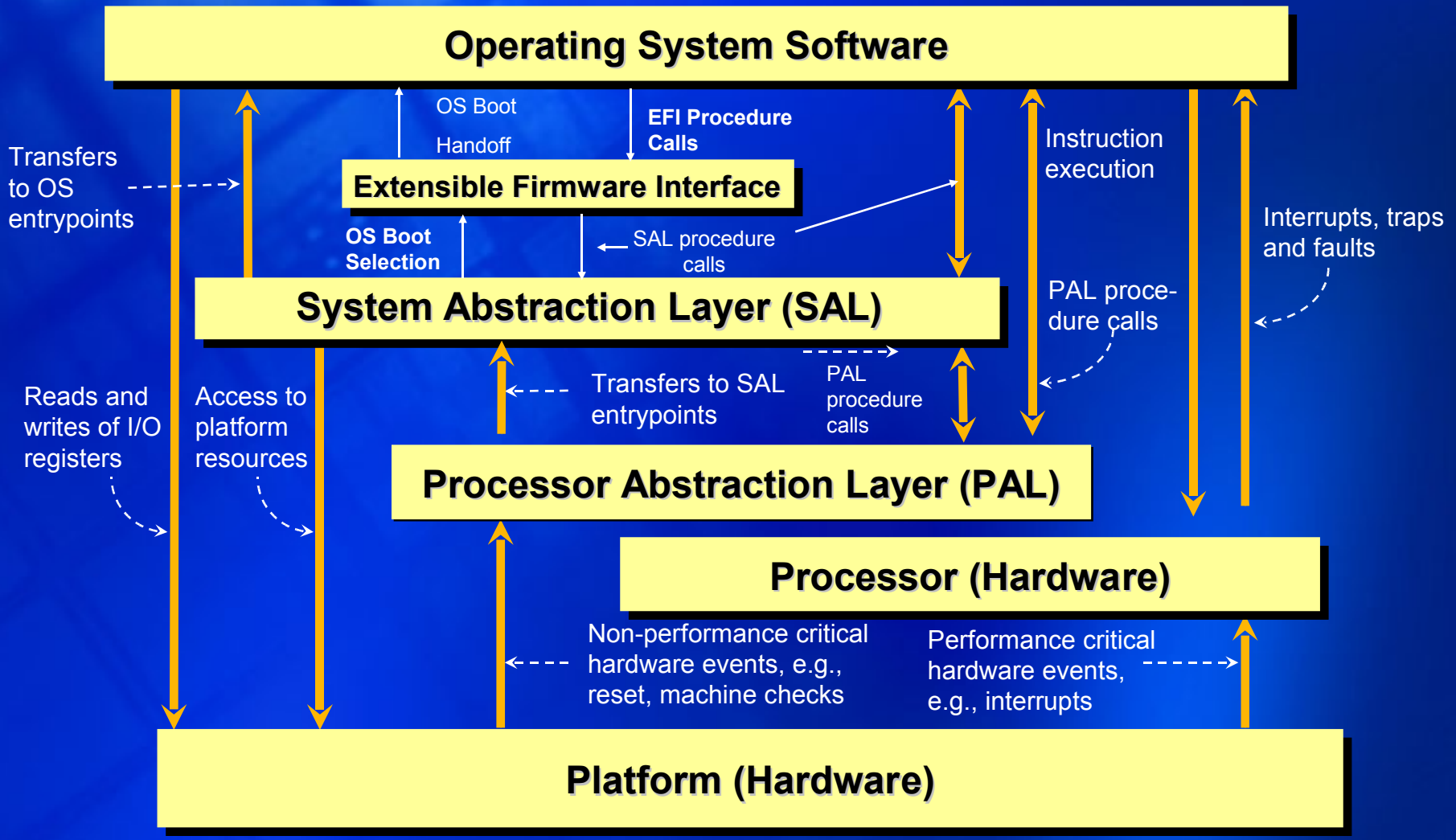
Agenda

- Overview of Itanium® Architecture PAL Firmware
- PAL calls
  - What they do
  - How Linux uses them
- Itanium Architecture Firmware Flows
  - Reset sequence
  - Machine check abort

Why is Itanium® Architecture-Based Firmware Different?

- IA-32 firmware
  - Developed in an ad-hoc manner
  - No explicit IA-32 firmware architecture
  - Loosely defined interfaces
- Itanium® architecture-based firmware
  - Explicit architecture
  - Abstracts changes in processor and platform implementations

Itanium® Architecture-Based Firmware



Processor Abstraction Layer

- PAL provides a consistent interface to processor features
  - PAL interface is defined in the *Intel® Itanium® Architecture Software Developer's Manual*

PAL Functionality

- Implements non-performance critical processor functions
  - Simplifies hardware implementation
  - Allows silicon to be used for performance enhancements
- PAL functions
  - Processor configuration
  - Processor feature identification
  - Processor self-test and initialization
  - PAL machine check error handling
  - Processor power management
- Intel provides PAL binary
  - PAL authentication prevents PAL modifications
  - PAL version checking requires a stepping-appropriate PAL release

PAL Components

- PAL_A
  - Entry points for Reset, Init, and machine check
  - Minimal PAL procedures for recovery
  - Stepping independent
    - Errata workarounds that may be provided in PAL_A
    - PAL_A is revised during development
  - May be placed in protected boot block
- PAL_B
  - Entry point for PMI
  - Reset, Init, PMI, and machine check handlers
  - Remaining PAL procedures
  - Stepping dependent
  - Production PAL supports all previous production steppings

PAL and SAL vs. Processor Steppings

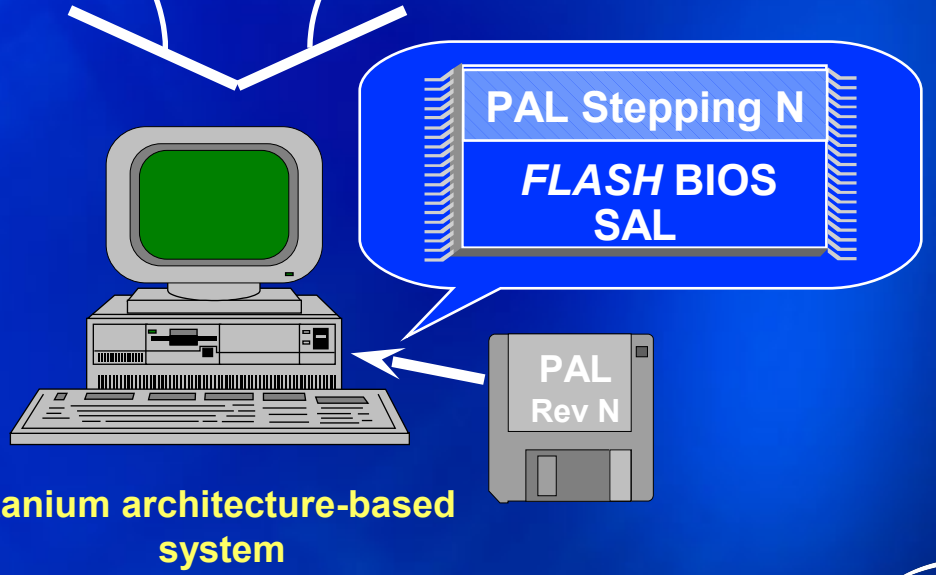
Itanium® Processor Stepping N

Itanium Processor Stepping N-1

Itanium Processor Stepping N-2

• • •

- Production PAL supports all production processor steppings
- PAL_B update/recovery may be required to use new processor stepping
- SAL should be independent of processor steppings



PAL Entrypoints

- Reset
  - Power-on and warm reset
- Init
  - Typically used for “crash dump”
  - If supported by OS, Init flow is continuable
- Machine Check Abort
  - Machine checks requiring PAL, SAL, or OS intervention
- PMI
  - Replaces IA-32 System Management Mode (SMM)
- Virtualization
  - See PAL_VM_* calls

PAL Procedures

- Provides APIs to processor-specific functionality
- Calling convention
  - All PAL calls may be made in physical mode
  - Some PAL calls may be called in virtual mode
  - SDM indicates whether PAL procedure is called using static or stacked registers
  - PAL calls may be made cacheable or uncacheable
- On following slides mark those used by Linux († = used, ‡ = used only in /proc)

PAL Info and Config Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_BUS_GET_FEATURES ‡	Return configurable processor bus interface features and their current settings
• PAL_BUS_SET_FEATURES	Enable or disable configurable features in processor bus interface
• PAL_DEBUG_INFO ‡	Return the number of instruction and data breakpoint registers
• PAL_FIXED_ADDR	Return the fixed component of a processor's directed address
• PAL_FREQ_BASE ‡	Return base frequency of platform if generated by processor
• PAL_FREQ_RATIOS†	Return the ratio of processor, bus and interval time counter to base platform frequency

PAL Info and Config Procedures (Continued)

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_PERF_MON_INFO†	Return the number of and type of performance monitors
• PAL_PLATFORM_ADDR	Specify processor interrupt block address and I/O port space address
• PAL_PROC_GET_FEATURES†	Return configurable processor features and their current settings
• PAL_PROC_SET_FEATURES	Enable or disable configurable processor features
• PAL_REGISTER_INFO ‡	Return AR and CR register information
• PAL_RSE_INFO†	Return RSE information
• PAL_VERSION ‡	Return the MINIMUM PAL version needed by the processor

PAL Info and Config Procedures (Continued)

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_LOGICAL_TO_PHYSICAL†	Returns information on the logical to physical processor mapping
• PAL_CACHE_SHARED_INFO†	Returns information on caches shared between logical processors

PAL Machine Check Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_MC_CLEAR_LOG	Clear all error info from processor error logging registers
• PAL_MC_DRAIN†	Ensure that all operations that could cause a MCA have been completed
• PAL_MC_EXPECTED	Set/Reset expected machine check indicator
• PAL_MC_DYNAMIC_STATE	Return processor dynamic state for logging by SAL
• PAL_MC_ERROR_INFO	Return processor machine check info and processor static state for logging by SAL
• PAL_MC_RESUME	Restore minimal architected state and return to interrupted process
• PAL_MC_REGISTER_MEM	Register minimal state save area with PAL for machine checks and inits

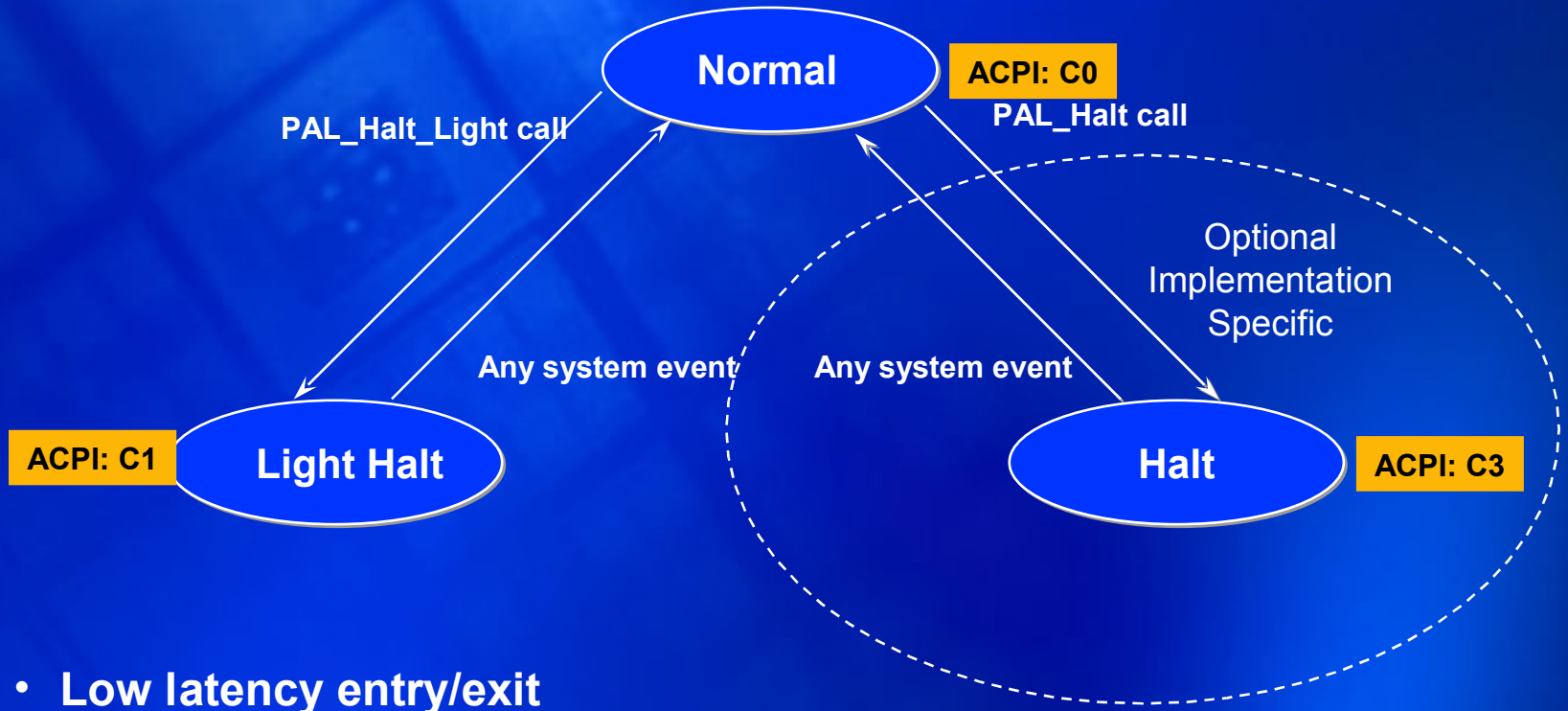
PAL Cache and Memory Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_CACHE_FLUSH†	Flush instruction or data caches
• PAL_CACHE_INFO†	Return detailed instruction or data cache information
• PAL_CACHE_INIT	Initialize the instruction or data caches
• PAL_CACHE_PROT_INFO	Return instruction or data cache protection information
• PAL_CACHE_SUMMARY†	Return a summary of the cache hierarchy
• PAL_MEM_ATTRIB ‡	Return a list of supported memory attributes
• PAL_PTCE_INFO†	Return info needed for PTCE instruction to purge entire TC
• PAL_VM_INFO ‡	Return detailed virtual memory features supported in the processor

PAL Cache and Memory Procedures
(Continued)

PAL PROCEDURE	*FUNCTIONALITY*
• **PAL_VM_PAGE_SIZE†**	**Return virtual memory TC and hardware walker page sizes supported in the processor**
• **PAL_VM_SUMMARY†**	**Return summary of virtual memory features supported in the processor**
• **PAL_VM_TR_READ ‡**	**Read contents of a translation register**

Itanium® Architecture Power (C-state) Management



- Low latency entry/exit
- Instruction execution stopped
- Cache coherency maintained

PAL Power Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• **PAL_HALT†**	**Enter the low power HALT state or an implementation dependent low power state**
• **PAL_HALT_LIGHT†**	**Enter low power Light HALT state**
• **PAL_HALT_INFO†**	**Return the low power capability of the processor**

PAL Diagnostic Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• PAL_CACHE_LINE_INIT	Initializes tags and data of a cache line for processor testing
• PAL_CACHE_READ	Read tag and data of a cache line for diagnostic testing
• PAL_CACHE_WRITE	Write tag and data of a cache for diagnostic testing
• PAL_TEST_INFO	Return the amount of memory needed for late processor self-test
• PAL_TEST_PROC	Perform late processor self-test

PAL Support Procedures

PAL PROCEDURE	*FUNCTIONALITY*
• **PAL_COPY_INFO**	**Return info needed to relocate PAL procedures and PAL PMI code to memory**
• **PAL_COPY_PAL**	**Relocate PAL procedures and PAL PMI code to memory**
• **PAL_PMI_ENTRYPOINT**	**Register PMI memory entrypoints with processor**

More PAL calls

PAL PROCEDURE	*FUNCTIONALITY*
• **PAL_PREFETCH_VISIBILITY†**	• **Change virtual memory attributes**

Newer PAL calls (in SDM 2.2)

- PAL_BRAND_INFO ‡
  - Processor branding information
- PAL_GET_HW_POLICY
- PAL_SET_HW_POLICY
  - H/w thread priority tuning
- PAL_GET_PSTATE†
- PAL_SET_PSTATE†
- PAL_PSTATE_INFO
  - Power (P-state) tuning
- PAL_MC_ERROR_INJECT†
  - Inject errors for testing
- PAL_MEMORY_BUFFER
  - Provide PAL a cacheable memory buffer
- PAL_SHUTDOWN
  - Put processor into low power state until RESET

PAL calls for Virtualization

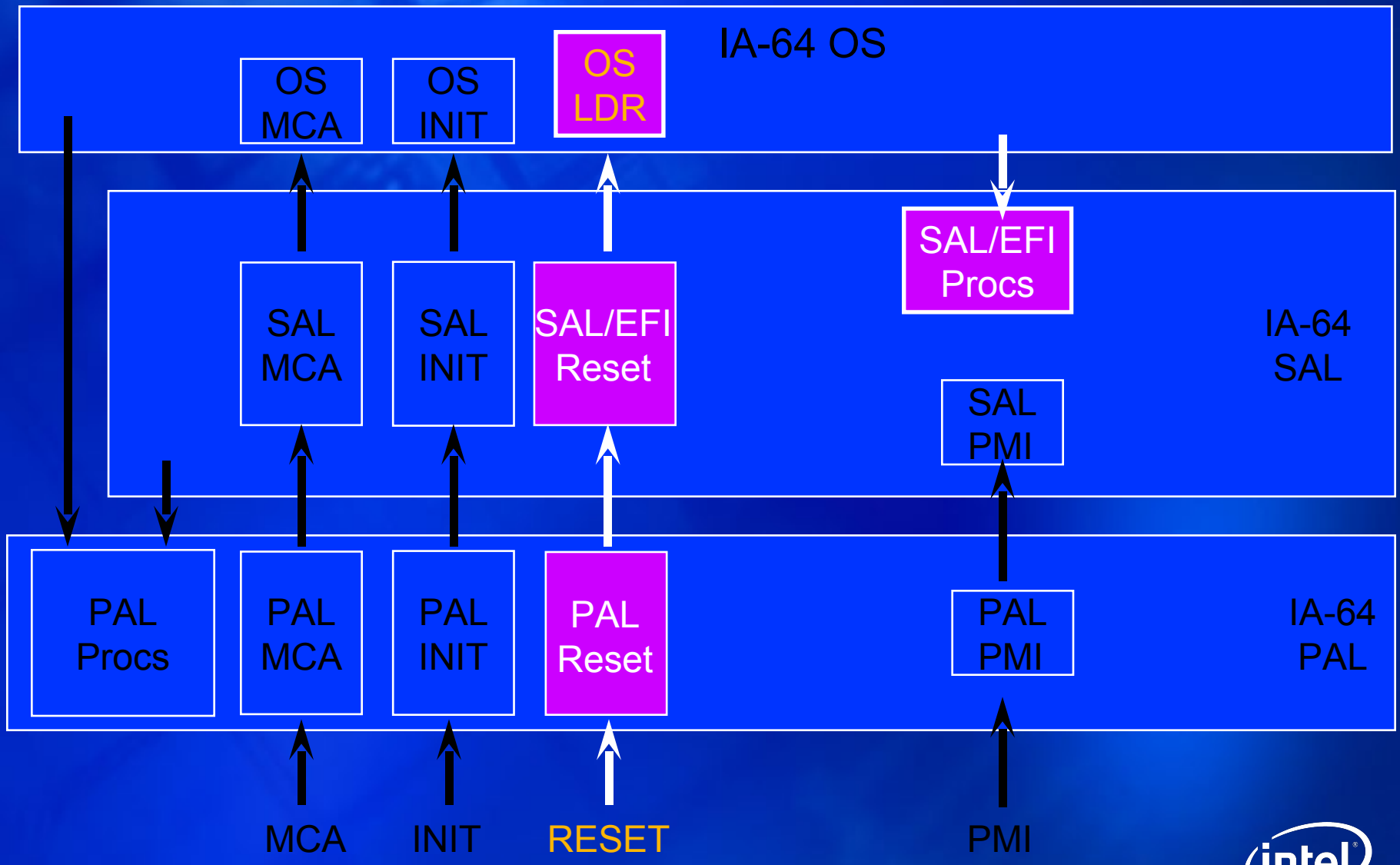
- PAL_VP_CREATE
- PAL_VP_ENV_INFO
- PAL_VP_EXIT_ENV
- PAL_VP_INIT_ENV
- PAL_VP_REGISTER
- PAL_VP_RESTORE
- PAL_VP_SAVE
- PAL_VP_TERMINATE

- Regular PAL calls to create, manipulate and terminate virtual processors

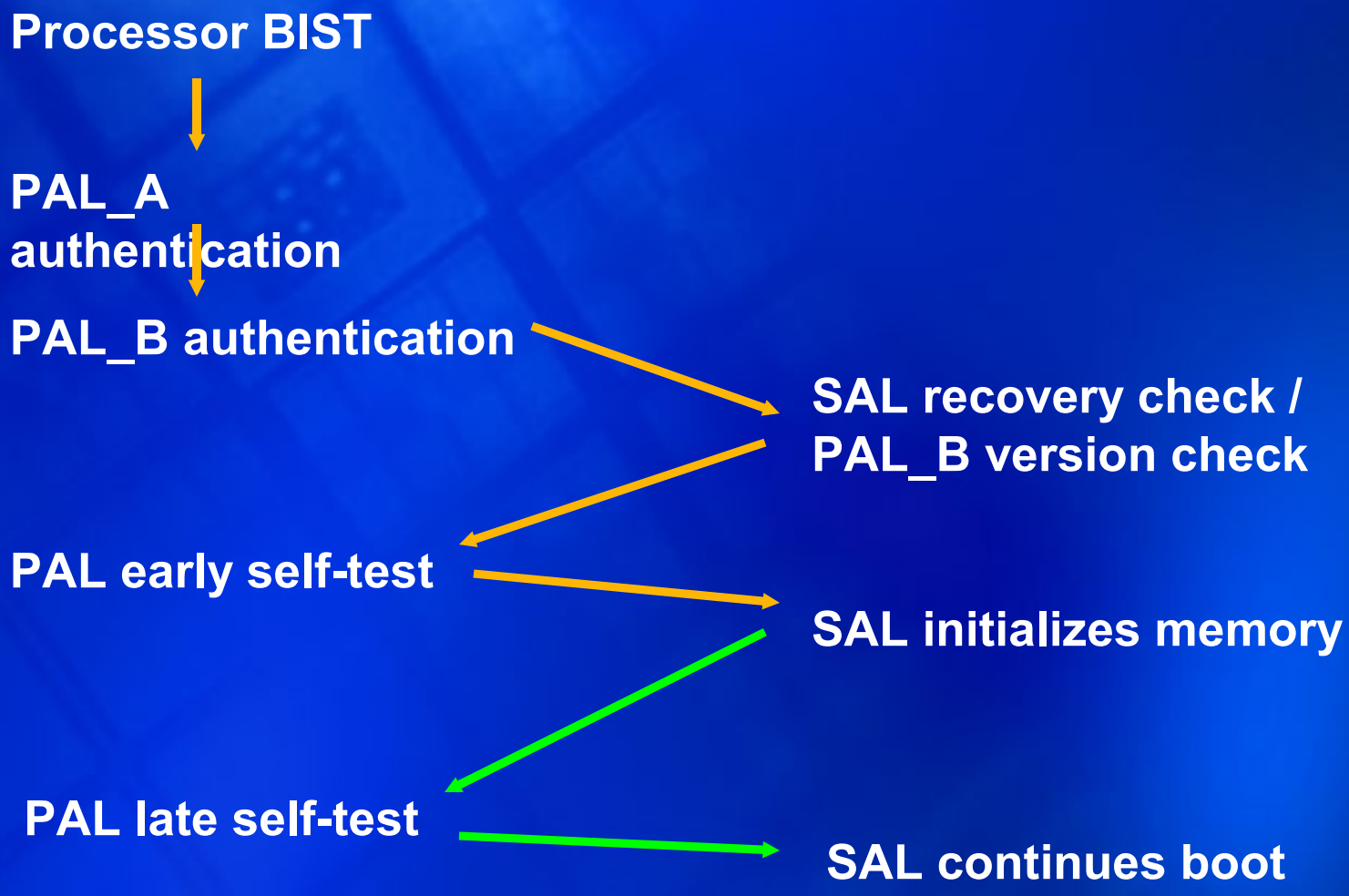
- PAL_VPS_*

- Virtualization entry points into VMM from PAL

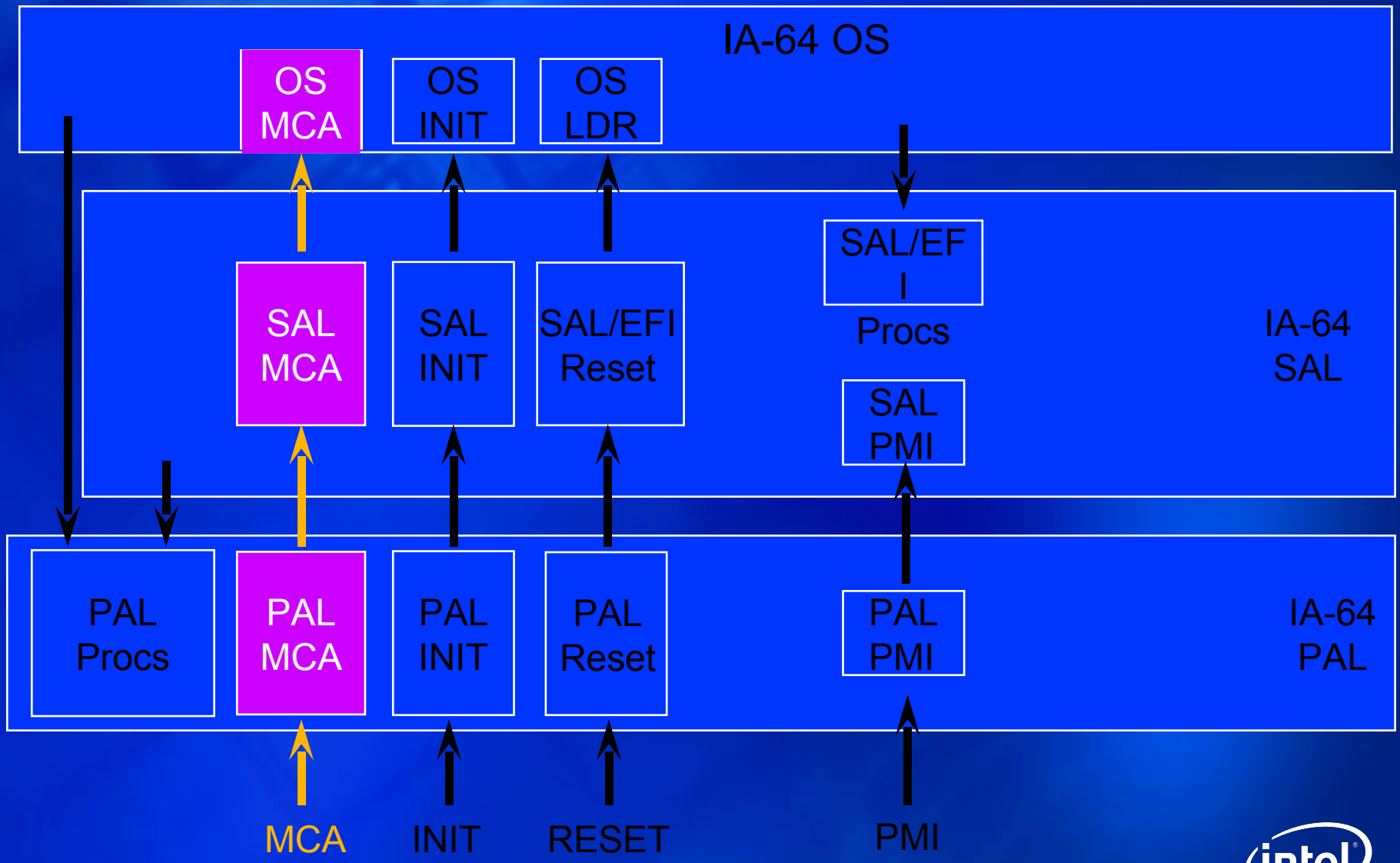
Itanium® Architecture Reset Sequence



Reset Sequence



Machine Check Abort Sequence




intel®
Leap ahead™